Peter van Ooijen is zelfstandig .NET consultant en ontwikkelaar

Software ontwikkelen voor de tablet pc

DE MOGELIJKHEDEN VAN PENINVOER

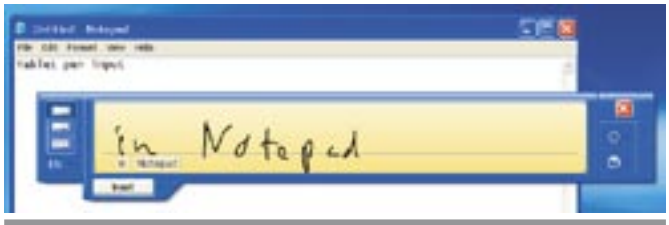
De tablet pc is een krachtig concept waarmee je zeer productief kunt zijn. In vele branches waar mensen tot nu toe altijd rondliepen met een schrijfblok, neem bijvoorbeeld medici en verzekeringsadviseurs, is de tablet pc een groot succes. In #3 van het .NET Magazine is de tablet pc eerder in de schijnwerpers gezet. In dit artikel wil ik met name verder ingaan op de mogelijkheden van peninvoer.

De tablet pc is al jaren veelbelovend. De eerste modellen lieten de potentie van het concept zien, maar in de praktijk waren de machines te duur en te traag om het platform tot een groot succes te maken. Het laatste jaar is daar stapje voor stapje het nodige aan veranderd. Met de komst van het Intel Centrino-platform is voldoende krachtige mobiele hardware beschikbaar en bijna alle grote merken bouwen tegenwoordig tablet pc's. Zo heeft op het moment dat ik dit schrijf IBM net de Tablet Thinkpad aangekondigd en Motion Computing een zeer compact model van 23x17cm! Een tablet pc is nog steeds iets duurder dan een 'gewone' notebook, maar voor dat geld krijg je dan ook wat: een geïntegreerde digitizer met pen waarmee je je pc kunt gebruiken alsof het een stuk papier is.

Een belangrijke stap in de ontwikkeling van de tablet pc is het besturingssysteem. Tegelijk met Microsoft Windows XP SP2 kwam Windows XP Tablet PC Edition 2005 uit. Dit is een volwaardige versie van Windows XP Pro met als extra functionaliteit:

- De pen als muis
- De pen als tekenpen
- Handschriftherkenning
- Peninvoer voor alle applicaties. Overal waar je kunt typen verschijnt het tablet inputpanel als pop-up. Zie afbeelding 1 als voorbeeld.
- Penspecifieke applicaties zoals Windows Journal en InkBall

Het mooie van Windows XP Tablet Edition is dat het volledig geïntegreerd is met het 'normale' Windows. Peninvoer en handschriftherkenning kun je ook gebruiken in applicaties die nog nooit een tablet pc hebben gezien. Daarnaast kun je specifieke tablet-applicaties ook draaien op een niet tablet pc, daar neemt de muis de rol van de pen over. Alleen handschriftherkenning, het vertalen van geschreven tekst in getypte tekst, werkt slechts op een tablet pc. Opgemerkt moet worden dat deze herkenning taalspecifiek is.
In #3 van het .NET Magazine, van april 2003, heeft Gerrit Jan Straatsma al een inleiding gegeven op het ontwikkelen van tablet pc-applicaties. In dit artikel wil ik met name verder ingaan op de mogelijkheden van peninvoer die Gerrit Jan al heeft laten zien. We hoeven niet met niets te beginnen. Het is heel goed mogelijk tablet-functionaliteit toe te voegen aan bestaande applicaties. Van daar uit duiken we stap voor stap dieper in tablet-specifieke functionaliteit.



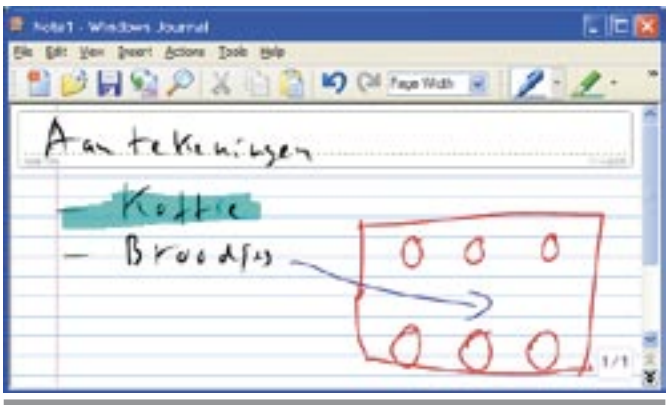
Afbeelding 1. Tablet inputpanel in Notepad

Tablet-functionaliteit toevoegen

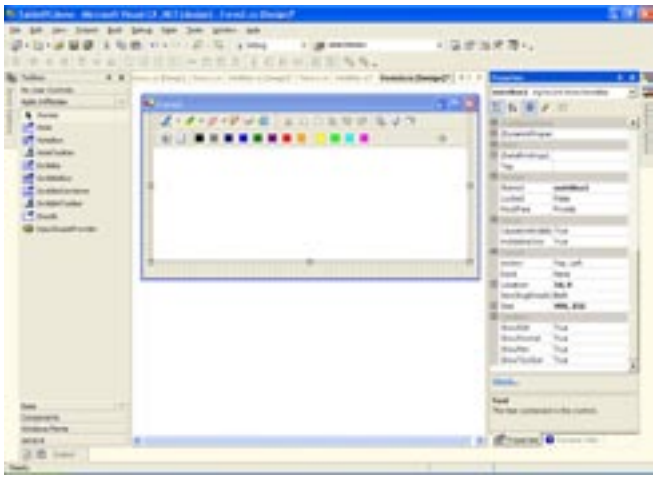
De interne implementatie van de tablet pc-API gebruikt COM. Naar de ontwikkelaar toe biedt de API een serie van COM-interfaces en een managed (.NET) wrapper. Dankzij de COM-interfaces kun je ook tablet-functionaliteit toevoegen aan bestaande applicaties die nog uit het pre-.NET-tijdperk stammen. Microsoft is weliswaar gestopt met de 'mainstream' support voor Visual Basic 6.0 (de 'extended' support loopt tot maart 2008), maar je kunt wel tablet-functionaliteit aan VB6-applicaties toevoegen. Gelukkig zijn er nog vele andere redenen om over te stappen op .NET als ontwikkelomgeving. Het enige echte verschil tussen de COM en de managed variant van de tablet-API is de syntax. In de rest van dit artikel werk ik met de managed API, maar alle code is dus net zo goed toe te passen in Visual Basic 6.0, Visual C++ of Delphi.

Agilix Infinotes: Windows Journal-functionaliteit toevoegen aan je applicatie

Windows Journal (in afbeelding 2 te zien) is een onderdeel van Windows XP Tablet PC Edition. Het is een volledige implementatie van gedigitaliseerd papier.
Op de blaadjes kun je aantekeningen maken en schrijven met pennen van variabele kleur en lijndikte. Wat je niet bevalt gum je weer



Afbeelding 2. Windows Journal



Afbeelding 3. Agilix Infinotes in Visual Studio

uit en als er geen ruimte meer is, schuif je het gekrabbel een eindje op. Met digitaal papier kan je meer dan met echt papier. Dit is allemaal functionaliteit die je graag binnen je eigen applicaties zou willen gebruiken. De tablet pc-API bevat twee basiscontrols waarmee het bouwen van een Journal-achtige control goed te doen is, maar eigenlijk zit je dan wel het wiel weer opnieuw uit te vinden. In nauwe samenwerking met Microsoft heeft Agilix (www.Agilix.com) de InfiNotes-controlsuite gemaakt. Dit is een serie windows-controls die je toevoegt aan de toolbox van Visual Studio; zie afbeelding 3. De suite komt in twee smaken: Standard Edition en Professional Edition. De Standard Edition is gratis (!), hiermee sleep je met één muisbeweging een compleet Journal-control op je winform. De control kan bestaande Windows Jounal note-files inlezen.

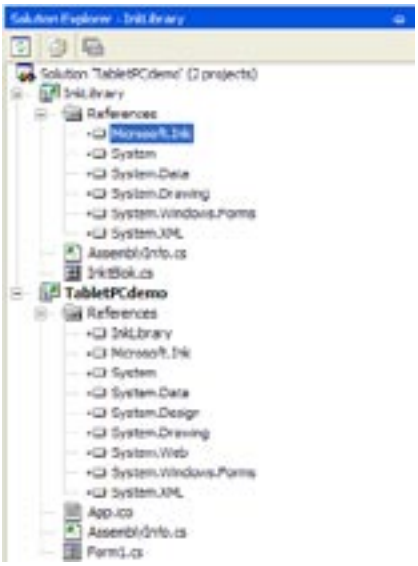
Ik ga hier niet verder in op het gebruik van de componenten, dat wijst zich vanzelf. Als je de volle functionaliteit van InfiNotes wil, is er de Professional Edition. Deze biedt zaken als geïntegreerde handschriftherkenning en notes van meer pagina's. Voor een ontwikkelaarlicentie van de Professional Edition betaal je een bescheiden bedrag. Er hoeven geen licentiekosten voor de distributie van gemaakte applicaties betaald te worden. Om het geld hoef je het dus niet te laten.

De tablet-API verkennen

Met de Infinotes-controls kan je snel en mooi beschrijfbare notitieblaadjes aan je applicatie toevoegen. Maar om de pen- en inktfunctionaliteit in andere delen van je applicatie te gebruiken zal je zelf moeten gaan programmeren. Op de Microsoft-site is de Tablet PC SDK te downloaden. Die bevat de volledige documentatie van de tablet-API-klassen en een aantal voorbeelden. Deze laatste springen een beetje van de hak op de tak, verschillende programmeertalen en stukjes van de API worden niet erg samenhangend beschreven. Een hele goede bron is *Building Tablet PC Applications* door Philip Su en Rob Jarret (Microsoft Press, ISBN 0-7356-1723-6). Dit is het enige boek over tablet pc-softwareontwikkeling op de markt. Het behandelt alleen de basis van de API, maar bevat een schat aan heldere informatie. In de rest van dit artikel zal ik de basis van de tablet-API doorlopen aan de hand van een klein demoproject; zie afbeelding 4. Het wordt een blaadje om op te kliederen. De applicatie zal zowel getekende vormen als geschreven tekst herkennen. Het tekenblaadje wordt uitgevoerd als Windows User control. Dat kan je gebruiken op een Windows-form en tot slot zal ik laten zien dat je het met een paar regels code ook op het web kunt gebruiken.

In dit verhaal zal ik een aantal zaken demonstreren:

- Met de pen tekenen op een Windows-form.
- De getekende inkt manipuleren.
- Specifieke penbewegingen herkennen.
- Handschriftherkenning.



Afbeelding 4. Demoproject Solution

De solution bestaat uit een Windows Forms-applicatie en een class library. Aan de class library voeg ik het Windows User control toe. Aan de references van de class library wordt de tablet pc-API dll toegevoegd. De Microsoft.Ink namespace bevat alle managed types van de tablet-API.

Deze namespace bevat een groot aantal types. De belangrijkste klassen die in de rest van dit verhaal voorbijkomen om bovengenoemde functionaliteit te realiseren zijn:

- **InkOverlay.** Een object van deze klasse reageert op de bewegingen van de pen. Deze bewegingen worden herkend als specifieke beweging (een 'gesture') of getekend als inkt (in 'strokes'). Een inkoverlay-object stelt de strokes en gestures beschikbaar aan je code via events en properties.
- **Stroke(s).** De getekende inkt wordt opgeslagen in een collectie van strokes. Stroke-objecten bevatten een verwijzing naar getekende pennenstreken.
- **Gesture(s).** De pen heeft een specifieke beweging gemaakt zoals het tekenen van een vierkant. Bij het herkennen van een gesture vuurt de inkoverlay een event af waar je code op kan reageren.
- **Recognizer.** Een recognizer-object interpreteert de verzamelde strokes als handschrift en kan dit omzetten naar tekst.

Ink verzamelen in strokes

De door de pen geschreven inkt is in Windows een volwaardig datatype. Het wordt verzameld als een verzameling van zogeheten *Strokes*, dat zijn pennenstreken. Een *stroke* is een verzameling van punten waardoor een vloeiende lijn loopt. De plaatselijke dikte van de lijn is afhankelijk van de druk van de pen op de digitizer. Het interne pen-inputsysteem verzorgt het verzamelen van punten en druk en tekent er een vloeiende lijn door.
In codevoorbeeld 1 wordt in de constructor *InktBlok* de peninvoer geïnitialiseerd. Allereerst moeten we iets hebben om op te kunnen schrijven. Dit wordt een gewoon panel. De Ink-API kan op elk Windows-control met een Windows-handle tekenen. Een *Microsoft.Ink.InkOverlay*-object gaat de peninvoer verzamelen in een collectie van *Strokes*. Zoals de naam van de klasse al suggereert wordt een *InkOverlay*-object bij wijze van spreken over het panel heengelegd. In de constructor van de user control wordt het object aangemaakt, de constructor van de *InkOverlay*-klasse kent een groot aantal overloads. Er is er één die een Windows-handle accepteert en ook één die een control accepteert. De gekozen overload van de constructor bepaalt het niveau van code access security dat de user control vereist om geladen te kunnen worden. Soms heb je alleen een handle naar het control waarop je wil gaan tekenen, bijvoorbeeld een ActiveX-control. Een Windows-handle is voor .NET-code een

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Windows.Forms;
using Microsoft.Ink;
namespace InkLibrary
{
   /// <summary>
   /// Summary description for InktBlok.
   /// </summary>
   public class InktBlok : System.Windows.Forms.UserControl
   {
      private System.Windows.Forms.Panel panel1;
      private Microsoft.Ink.InkOverlay ic;
      private System.Windows.Forms.TabControl tabControl1;
      private System.Windows.Forms.TabPage tabPage1;
      private System.Windows.Forms.TabPage tabPage2;
      private System.Windows.Forms.Label labelApplicationGesture;
      private System.Windows.Forms.Button buttonWisStrokes;
      private System.Windows.Forms.CheckBox checkBoxLaatsteStroke;
      private System.Windows.Forms.Label labelSystemGesture;
      private System.Windows.Forms.CheckBox checkBoxApplicationGestures;
      private System.Windows.Forms.TextBox textBoxVoorKeurwoord;
      private System.Windows.Forms.ListBox listBoxRecoResultaat;
      private System.Windows.Forms.Label label1;
      private System.Windows.Forms.Button buttonHerkenHandschrift;
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.Container components = null;

      public InktBlok() {
         // This call is required by the Windows.Forms Form Designer.
         InitializeComponent();
         // Het ic object verzamelt de Ink Strokes en tekent ze op panel1
         ic = new InkOverlay(panel1);
         // Ontvang notificaties van system-gestures
         ic.SystemGesture+=new InkCollectorSystemGestureEventHandler
            (ic_SystemGesture);
         // Ontvang notificaties van application gestures
         ic.Gesture +=new InkCollectorGestureEventHandler(ic_Gesture);
         ic.CollectionMode = CollectionMode.InkAndGesture;
         ic.Enabled = true;
      }

      protected override void Dispose( bool disposing ) {
         if( disposing )
         {
            if(components != null)
            {
               components.Dispose();
               if (ic != null)
                  ic.Dispose();
            }
         }
         base.Dispose( disposing );
      }

      public void DisposeResources() {
         Dispose(true);
      }

      private void menuItemKleur_Click(object sender, System.EventArgs e) {
         colorDialog1.Color = ic.DefaultDrawingAttributes.Color;
         if (colorDialog1.ShowDialog() == DialogResult.OK)
            ic.DefaultDrawingAttributes.Color = colorDialog1.Color;
      }

      private void menuItemDruk_Click(object sender, System.EventArgs e) {
         ic.DefaultDrawingAttributes.IgnorePressure =
            ! ic.DefaultDrawingAttributes.IgnorePressure;
         menuItemDruk.Checked = ! ic.DefaultDrawingAttributes.IgnorePressure;
      }

      private void menuItemInkt_Click(object sender, System.EventArgs e) {
         ic.EditingMode = InkOverlayEditingMode.Ink;
      }

      private void menuItemSelecteer_Click(object sender, System.EventArgs e) {
         ic.EditingMode = InkOverlayEditingMode.Select;
      }

      private void menuItemWisPunt_Click(object sender, System.EventArgs e) {
         ic.EditingMode = InkOverlayEditingMode.Delete;
         ic.EraserMode = InkOverlayEraserMode.PointErase;
      }

      private void menuItemWisStreek_Click(object sender, System.EventArgs e) {
         ic.EditingMode = InkOverlayEditingMode.Delete;
         ic.EraserMode = InkOverlayEraserMode.StrokeErase;
      }

      private void buttonWisStrokes_Click(object sender, System.EventArgs e) {
         // Wacht tot alle peninvoer verwerkt is
         while (ic.CollectingInk);
         // Alle strokes wissen ?
         if (! checkBoxLaatsteStroke.Checked)
            ic.Ink.DeleteStrokes();
         else
         {
            // Alleen laatste stroke wissen
            // Maak een nieuw (refrentie naar) Strokes collectie aan
            Microsoft.Ink.Strokes sc = ic.Ink.CreateStrokes();
            // Voeg hier de laatst getekende stroke aan toe
            if (ic.Ink.Strokes.Count > 0)
               sc.Add(ic.Ink.Strokes[ic.Ink.Strokes.Count -1]);
            // Wis de strokes in de collectie
            ic.Ink.DeleteStrokes(sc);
         }
         panel1.Invalidate();
      }

      private void ic_SystemGesture(object sender,
         InkCollectorSystemGestureEventArgs e) {
         labelSystemGesture.Text = e.Id.ToString();
      }

      private void checkBoxApplicationGestures_CheckedChanged(object sender,
         System.EventArgs e) {
         ic.SetGestureStatus(ApplicationGesture.AllGestures,
            checkBoxApplicationGestures.Checked);
      }

      private void ic_Gesture(object sender, InkCollectorGestureEventArgs e) {
         labelApplicationGesture.Text = "";
         Rectangle rcBounds = e.Strokes.GetBoundingBox();
         for (int i = 0; i < e.Gestures.Length; i++)
         {
            labelApplicationGesture.Text+=   e.Gestures[i].Id.ToString() + ";";
            if (e.Gestures[i].Id == ApplicationGesture.Square)
            {
               Point[] pts = new Point[5];
               pts[0] = new Point(rcBounds.Left, rcBounds.Top);
               pts[1] = new Point(rcBounds.Right, rcBounds.Top);
               pts[2] = new Point(rcBounds.Right, rcBounds.Bottom);
               pts[3] = new Point(rcBounds.Left, rcBounds.Bottom);
               pts[4] = pts[0];
               Stroke str = ic.Ink.CreateStroke(pts);
               switch(e.Gestures[i].Confidence)
               {
                  case RecognitionConfidence.Strong :
                     str.DrawingAttributes.Color = Color.Green;
                     break;
                  case RecognitionConfidence.Intermediate :
                     str.DrawingAttributes.Color = Color.Orange;
                     break;
                  case RecognitionConfidence.Poor :
                     str.DrawingAttributes.Color = Color.Red;
                     break;
               }
               ic.Ink.Strokes.Add(str);
            }
         }
      }
      private void buttonHerkenHandschrift_Click(object sender,
         System.EventArgs e) {
         while (ic.CollectingInk);
         Strokes handSchrift = ic.Selection;
         // Is er iets geschreven ?
         if (handSchrift == 0)
         {
            MessageBox.Show("Geen inkt gevonden");
            return;
         }
         // Draait de code op een tablet PC ?
         Recognizers mijnTalen = new Recognizers();
         if (mijnTalen.Count == 0)
         {
            MessageBox.Show("Geen handschriftherkennig geinstalleerd");
            return;
         }
         // Zoek de recognizer voor en-us
         // De default haalt de locale op vogens datum-tijd formaat
         int lcid = 0x0409;
         Recognizer mijnTaal = mijnTalen.GetDefaultRecognizer(lcid);
         RecognizerContext mijnContext = mijnTaal.CreateRecognizerContext();
         // Voorkeur voor resultaat
         WordList mijnWoorden = new WordList();
         if (textBoxVoorKeurwoord.Text != "")
         {
            mijnWoorden.Add(textBoxVoorKeurwoord.Text);
            mijnContext.WordList = mijnWoorden;
         }
         // Herkennen tekst
         mijnContext.Strokes = handSchrift;
         RecognitionStatus mijnStatus;
         RecognitionResult myResult = mijnContext.Recognize(out mijnStatus);
         if (mijnStatus == RecognitionStatus.NoError)
         {
            RecognitionAlternates mijnAlternates =
               myResult.GetAlternatesFromSelection();
            listBoxRecoResultaat.Items.Clear();
            foreach (RecognitionAlternate ra in mijnAlternates)
               listBoxRecoResultaat.Items.Add(string.Format("Resultaat : {0},
      Kans dat het klopt: {1}", ra.ToString(), ra.Confidence.ToString()));
         }
         mijnContext.Dispose();
      }
      #region Component Designer generated code
      /// Weggelaten
      #endregion
   }
}
```

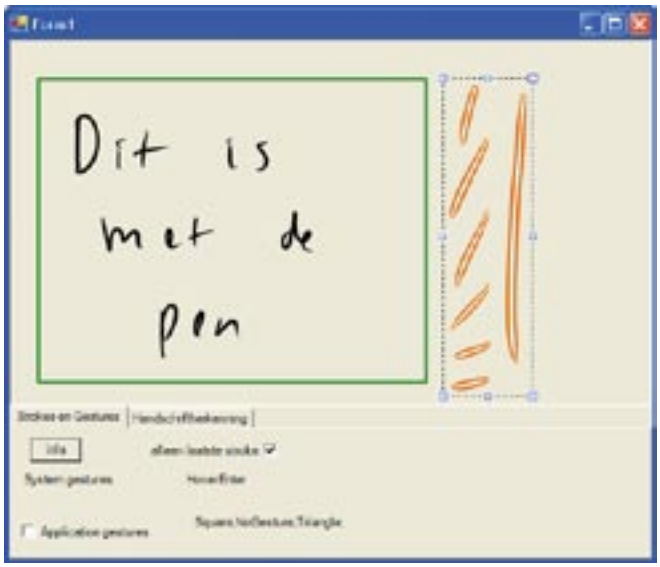
Codevoorbeeld 1.

externe resource. Het resultaat zal zijn dat deze class-library een te hoog security-level vereist om zonder meer door Internet Explorer geladen te worden. Geef je een control mee aan de constructor van de *InkOverlay* dan zakt het vereiste security-level en kunnen we de assembly straks zonder security-issues laden in Internet Explorer.

Door de enabled property van het InkOverlay-object op true te zetten kun je nu op het panel tekenen. De Ink-API zorgt ervoor dat de streken op het panel worden getekend . Als je de user control op een Windows-form sleept heb je een applicatie waar je op het panel kunt tekenen. Het tekenen van de inkt is vanuit je code in detail aan te sturen via de *DefaultDrawingAttributes* van de inkoverlay. Dit object heeft een groot aantal properties, een aantal ga ik instellen vanuit het contextmenu van de panel. Door op de rechtermuisknop te klikken, of om geheel in tablet-stijl te blijven, door een *press-and-hold* (druk met de pen en houdt hem even vast) verschijnt het menu. In het menu heb ik een aantal items geïmplementeerd. In *menuItemKleur_Click* wordt een tekenkleur gekozen door een standaard Windows-colordialog. Het tweede menu-item, *menuItemDruk_Click*, zet de *IgnorePressure*-property. Normaal resulteert harder drukken op de pen in een dikker getekende lijn. Als je de *IgnorePressure*-property uitzet, is de getekende lijn altijd even dik. Naast het tekenen van inkt kent de pen nog twee modi: selecteren en wissen. In selecteer-modus teken je met de pen een lasso om pennenstreken heen. De geselecteerde pennenstreken lichten op en er wordt een kader omheen getekend met handles. De streken kun je nu verslepen en vervormen door aan de handles te trekken. Al deze functionaliteit is ingebouwd onderdeel van de tablet-API.
De tablet-API kent twee wismodi: *PointErase* en *StrokeErase*. Voor beide is een menu-item opgenomen. In erase-modus verandert de cursor in een gummetje. In PointErase-modus werkt die als een gewone gum. Als je een stukje van een lijn wist, dan wordt in het codemodel van de strokes (dat we zo tegen zullen komen bij het programmatisch wissen van streken) een pennenstreek in stukken geknipt. Wat eerst één streek was, zijn nu twee (of meer) streken geworden. In *StrokeErase*-modus wist de pen veel rigoureuzer. Als je nu een pennenstreek met de gum aanraakt, wordt de hele streek in één keer gewist. Alles wat je in één haal had getekend verdwijnt weer. In het *InkOverlay*-object komt de verzamelde inkt terecht in de *Ink*-property, dit is een collectie van Strokes. Je wist de inkt programmatisch door deze Strokes-collectie weer leeg te maken. In codevoorbeeld 1 gebeurt dat in de *ButtonWisStrokes_Click*-method. Eerst wacht de lus *CollectingInk* tot alle peninvoer verwerkt is. Daarna verwijdert de *DeleteStrokes*-methode de strokes. Door het panel te invalideren met de *Invalidate*- methode wordt een repaint afgedwongen. Het bevat nu geen strokes meer en na het klikken van de knop zal het weer leeg zijn.

De collectie van *Strokes* bevat niet de inkt zelf maar een referentie naar inkt *Stroke*-objecten. Om een beter idee te krijgen van strokes en strokes-collecties ga ik nu alleen de laatst getekende stroke wissen. De *DeleteStrokes*-methode heeft een overload die een collectie strokes verwacht. Alle strokes in deze collectie worden gewist.
Je maakt een nieuwe strokes-collectie aan met de *CreateStrokes*-methode van de strokes-klasse. Deze initieert een nieuwe collectie strokes. De code voegt aan deze nieuwe collectie de laatste stroke uit de Ink-property toe en geeft de nieuwe collectie door aan de *DeleteStrokes*-methode. Je ziet weer dat een stroke één lijn is, getekend tussen de plaats waar je de pen hebt neergezet en waar je hem weer hebt opgetild, ongeacht het aantal krullen, lussen en bochten in de pennenstreek. Deze lijn wordt in één keer gewist. Elke klik van de knop laat een stroke verdwijnen.

Bij het sluiten van de form zelf is er nog één ding waar je goed rekening mee moet houden. We benaderen de tablet-API weliswaar via een managed class, maar niet alle resources die deze class gebruikt zijn managed. Dat betekent dat je het *InkOverlay*-object ook weer moet opruimen door zijn *Dispose*-method aan te roepen.



Afbeelding 5. De demoapplicatie aan het werk

Dit laatste wordt heel vaak vergeten, zelfs in Windows XP Tablet PC Edition. Microsoft heeft erkend dat daar een geheugenlek in zit. In de praktijk betekent dit dat je de machine eens in de zoveel tijd (dagen, zo niet weken) opnieuw moet opstarten. Inmiddels is er een patch voor dit lek beschikbaar. De *Dispose*-method is een onderdeel van het dispose pattern, hier roept elke container de methode aan van de componenten die het bevat. Een Windows-form doet dat op alle controls die er op staan; je ziet dat Visual Studio een *Dispose*-methode genereert voor een nieuw user control. Hierin moeten we de InkCollector gaan opruimen. Aangezien de control later ook in een niet .NET-host gebruikt gaat worden (Internet Explorer) krijgt de user control ook een publiek toegankelijke methode *DisposeResources* die de resources opruimt.

Gestures

De Tablet API werk met inkt als een verzameling *Strokes*. Maar de API gaat verder. Naast de getekende inkt herkent de API ook bewegingen van de pen. De pen kent een uitgebreid scala aan bewegingen. Hij tikt het scherm aan, maakt een sleepbeweging, tekent een cirkel en zo meer. Deze bewegingen worden intern door het pen-inputsysteem herkend. In de code kun je reageren op deze bewegingen. Dit zijn de zogeheten *Gestures*. Ze vallen uiteen in twee categorieën, System Gestures en Application Gestures. Door een eventhandler aan de *InkOverlay* aan te bieden, ontvangt je code een notificatie van de gesture. De eventhandler *ic_SystemGesture* krijgt specifieke eventargs aangeboden. Hierin vind je de ID van de geconstateerde gesture. De voorbeeldcode in *ic_SystemGesture* schrijft een tekstrepresentatie van deze gesture naar een label. Je zult zien dat een *Tap*-gesture optreedt als de pen het scherm raakt en een *Drag*-gesture als de pen over het scherm sleept. Heel interessant zijn de *HoverEnter*- en *HoverLeave*-gestures. Deze treden op voordat de pen het scherm aanraakt. Zodra de pen boven het panel hangt vuurt de gesture een event. De digitizer van een tablet pc reageert op verstoring van het magnetisch veld, dat overigens al wordt verstoord als de pen in de buurt komt. Bij normaal gebruik zie je ook dat je de cursor van een tablet kunt verplaatsen zonder het scherm te hoeven aanraken. En dat zie je hier dus ook terug in je code.

Naast de SystemGestures zijn er de ApplicationGestures. De gesture-recognizer is een ingebouwde vormenherkenner. Om een notificatie van een SystemGesture te ontvangen was het voldoende om een eventhandler te zetten. Default wordt geen enkele application-gesture herkend. Om de vormenherkenner aan te zetten, moet je eerst de CollectionMode-property van de InkOverlay op InkAndGesture zetten. Deze property staat default op *InkOnly*, naast *InkAndGesture* kent die ook nog *GestureOnly*. Om een noti-

ficatie van een specifieke ApplicationGesture te krijgen, moet je expliciet aangeven van welke gestures je genotificeerd wilt worden. Welke vormen herkend worden staat in de *ApplicationGestures*-enumeratie. In de democode wil ik in eerste instantie alle gestures zien en daarom zet ik hem op *ApplicationGestures.AllGestures*. In productiecode moet je dit niet doen, het herkennen van gestures is een kostbare zaak. Een volledige lijst van herkenbare gestures staat in de screenshot in afbeelding 6.

Werkend met de demo zul je zien dat een eenmaal herkende gesture niet getekend wordt. Eén van de vele gestures is *Down*. In de praktijk betekent dit dat het niet meer zal lukken een letter *l* of *i* te schrijven aangezien die al voor een down-gesture zijn aangezien. In de demo werk ik dit uit voor de *square*-gesture. De code gaat een geïdealiseerde rechthoek tekenen. Dat doet die door zelf een aantal strokes aan te maken. De *CreateStroke*-methode heeft daarvoor een aantal *Points* nodig. Een rechthoek wordt gedefinieerd door vier hoekpunten. Om een volledige rechthoek in één lijn te tekenen heb je vijf punten nodig, waarbij het begin- en eindpunt elkaar overlappen. De *InkCollectorGestureEventArgs*-parameter van de *ic_Gesture*-eventhandler bevat een collectie strokes, dit zijn de pennenstreken die gezet zijn. Deze strokes hebben een *GetBoundingBox*-methode die de vier hoekpunten levert die de strokes omsluiten. Hier kan ik de hoekpunten van de geïdealiseerde rechthoek zo uit halen. Van elke herkende gesture is ook de *Confidence* te lezen, die geeft weer hoeveel vertrouwen de recognizer in het resultaat heeft. Het niveau van vertrouwen bepaalt de kleur van de rechthoek. Als ik nu de zelf gemaakte strokes toevoeg aan de *Ink.Strokes* van de InkOverlay verschijnen ze in het panel. Voor de InkOverlay zijn deze strokes niet meer te onderscheiden van de met de hand getekende. Ze laten zich dan ook net zo gemakkelijk wissen als de met een pen getekende strokes.

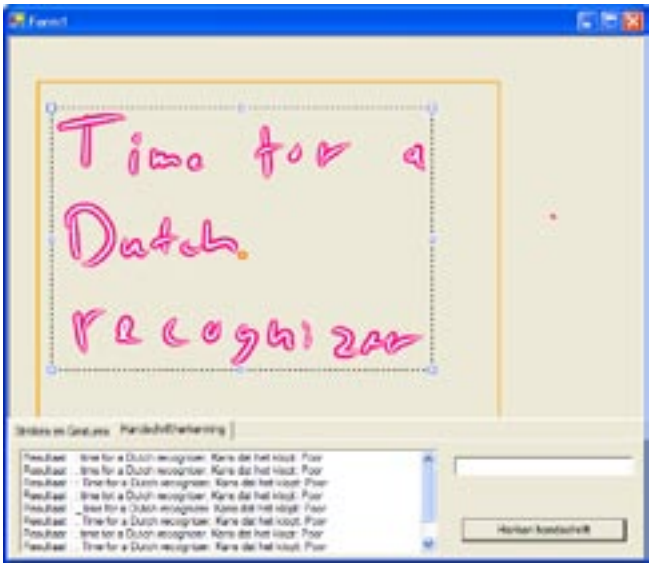
Afbeelding 6. Enumeratie application-gestures

Handschriftherkenning

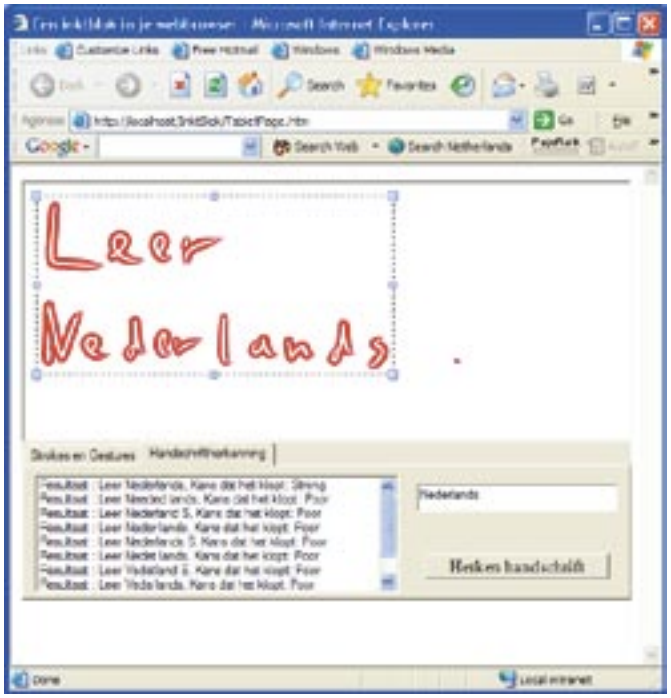
Naast tekenen kan je met een pen ook schrijven. De krabbels op het panel zouden net zo goed voor geschreven tekst kunnen staan.

Handschriftherkenning is een onlosmakelijk onderdeel van Windows XP Tablet PC Edition. Deze herkenning gebeurt op een intelligente manier. Op een PDA onder Windows Mobile moet je letter voor letter tekenen in een aangewezen vakje waarna het besturingssysteem de tekst letter voor letter probeert te herkennen. Het tabletbesturingssysteem pakt het op een slimmere manier aan, het herkent handschrift woord voor woord. Het nadeel is dat de handschriftherkenners (recognizers) daardoor taalgebonden zijn. Een voordeel is dan weer dat de herkenning niet gebonden is aan een specifieke karakterset. Van de momenteel beschikbare recognizers is er een aantal voor Chinese en Japanse talen. Een herkenner voor Nederlands is in ontwikkeling. Handschriftherkennning is vanuit de API goed bereikbaar. De snelste manier is de ToString-methode van de *Strokes*-klasse. Deze geeft default de waarschijnlijkste tekst terug. De basis van handschriftherkenning in je code is zo een one-liner. Maar niemand schrijft perfect, in de praktijk mist je code zo wel het één en ander. Het Tablet Input Panel van Windows XP Tablet PC Edition laat altijd een aantal alternatieven zien. Woorden die in zijn woordenboek staan, hebben de voorkeur. Maar mocht je nou een woord hebben geschreven dat daar niet in staat en je hebt ook redelijk netjes geschreven, dan zal je het geschrevene toch in de lijst van alternatieven tegenkomen. De tablet-recognizer werkt weliswaar op basis van een woordenlijst, maar hij beperkt zijn keuzes daar niet toe.

Deze alternatieven kun je ook vanuit je code benaderen, maar dat vraagt om enig programmeerwerk. De code voor handschriftherkeening vind je in codevoorbeeld 1 in de *buttonHerkenHandschrift_Click*-methode. Deze gaat de geselecteerde Strokes, die in de property *Selection* van de inkoverlay staan, als handschrift analyseren. In de eerste plaats moet die controleren of er wel *Strokes* zijn om te herkennen. Doe je dat niet dan krijg je bij een lege Strokes-collectie een exceptie om je oren geslingerd. Vervolgens moet die controleren of er wel een recognizer is geïnstalleerd. Een recognizer zal niet zijn geïnstalleerd op een niet tablet pc. De code draait daar wel, maar de herkenning kan niet worden uitgevoerd. Met de *GetDefaultRecognizer*-methode haal je een recognizer op. In principe haalt de default (parameterloze) overload van deze methode de juiste recognizer op, maar die gedraagt zich iets anders dan je zou willen. In Windows kan je een taalinstelling opgeven voor je datumformaat en een standaardtaal. Zo gebruik ik een Engelstalige Windows met het Nederlandse datumformaat. Het vervelende van de methode uit de API is dat die naar het locale formaat van de datum kijkt en niet naar dat van de invoertaal. In mijn geval zoekt die dus naar de Nederlandse recognizer en die is (nog) niet op mijn tablet geïnstalleerd. In de praktijk moet je met een lcid-constante de te gebruiken recognizer kiezen. De recognizer heeft een zogeheten context. Eén van de mooie dingen die je hier kunt doen is een woordenlijst meegeven. Woorden die in deze lijst staan hebben de voorkeur bij het herkennen van het resultaat. Zo kun je de handschriftherkenning flink bijsturen. Na al deze controles en instellingen kan de recognizer aan het werk. In het resultaat staat ook een lijst met alle alternatieven. Per alternatief valt te lezen hoeveel vertrouwen de recognizer er in heeft dat het bewuste woord overeenkomt met de aangeboden strokes.



Afbeelding 7. Handschriftherkenning



Afbeelding 8. Pen-control in Internet Explorer

In een webapplicatie

Ook in een webapplicatie kun je tabletfunctionaliteit aanbieden, maar dan moet je wel Internet Explorer gebruiken. De nieuwste versies hiervan kunnen .NET-assemblies laden en een visuele component zoals een user control kunnen ze ook tonen. Dit heeft niets met ASP.NET te maken, het werkt op iedere fileserver; ook op de Unix-doos van jouw provider. Internet Explorer laadt een object van een .NET-klasse. De naam van de dll waarin die klasse zich bevindt en de naam van de klasse geef je mee in het *ObjectID*-attribuut van een HTML- object. Codevoorbeeld 2 laat de volledige HTML-code zien van de pagina en afbeelding 6 toont het resultaat in de browser. In de ObjectID staat de naam van de dll gevolgd door een # met daarna de gekwalificeerde naam van de user control, dat is de naam inclusief zijn namespace. De webserver zal de dll in dezelfde map gaan zoeken als de webpagina.

Aan dit scenario zijn wat security-issues verbonden. Internet Explorer laadt niet zomaar willekeurige .NET-code, dat hangt af van het door de code vereiste level. Dat level hangt weer af van de door de assembly gebruikte resources. Bij het aanmaken van het Inkoverlay-object hadden we de keuze of dit met een Windows-handle zou gaan werken of met een Windows-control. Een Windows-handle is voor de assembly een (potentiële) externe resource, de assembly zal in dat geval een hoger level vereisen. Internet Explorer kijkt ook naar de locatie van de te laden dll. Maar als de assembly geen externe Windows-handles of externe mappen gebruikt, zal IE met zijn default (Secure) instellingen van de tablet-control laden.

De toekomst van de tablet pc

Zoals ik al in inleiding opmerkte verloopt de opmars van de tablet pc langzaam maar zeker. Hij volgt niet echt het pad van de Visual Studio-versies. De nieuwste versie van de Tablet SDK, met het weinigzeggende nummer 1.7, kwam uit met Windows XP Tablet Edition 2005. Voor Visual Studio 2005 staat wat de tablet betreft weinig spectaculairs op het programma. De grote stap gaat Longhorn worden. In Longhorn wordt Tablet Ink een native datatype, je hoeft dus niet meer tablet-assemblies mee uit te rollen met je tabletapplicatie. Naar verluidt komt er een Tablet Edition-versie van Longhorn naast de 'gewone' Longhorn. Het is wel de vraag

```
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Een inktblok in je webbrowser</title>
<SCRIPT language=jscript>
  // Release inkoverlay resources
    function OnUnload()
    {
        InktBlok.DisposeResources();
    }
  </SCRIPT>
</HEAD>
<body>
<OBJECT id=InktBlok
title="Een pc met de tablet pc-API's (geïnstalleerd) is nodig om deze
  pagina te bekijken."
height="90%" width="100%"
classid=InkLibrary.dll#InkLibrary.InktBlok  VIEWASTEXT>
</OBJECT>
</body></html>
```

Codevoorbeeld 2.

of er nog notebooks zonder tabletfunctionaltiet worden verkocht tegen de tijd dat Longhorn uitkomt.

Naast notebooks heb je ook nog Windows Mobile. Dit groeit meer en meer naar standaard Windows toe. Op dit moment heb je al pc's die niet groter zijn dan een flinke PDA en daar draait Windows Tablet Edition op. Maar je mag het geen echte tablet pc noemen. Er is officieel vastgelegd waaraan een tablet moet voldoen, daar valt bijvoorbeeld ook de ctr-alt-del-knop onder. Ook typisch tablet is de magnetische digitizer. Het idee is dat je met je hand op het scherm moet kunnen leunen als je met de pen schrijft. Windows Mobile-apparaten hebben een drukgevoelig scherm. Als je daar opleunt, dan denkt het scherm dat je wat invoert. Om de zaken helemaal verwarrend te maken wordt recentelijk weer erg veel aandacht besteed aan touchscreens in Longhorn. Ik kan het ook niet meer overzien, maar hoe je ook met een pen op het scherm van je machine tekent, als ontwikkelaar kan je er erg mooie dingen mee doen. Scribble ze!

Peter van Ooijen (Gekko Software www.Gekko-Software.nl) is werkzaam als zelfstandig .NET-consultant en ontwikkelaar. Naast de dagelijkse ASP.NET-kost is de tablet pc een geliefd troeteldier. Ook op zijn weblog http://codebetter.com/blogs/peter.van.ooijen/ komt de tablet pc geregeld voorbij. Voor vragen en opmerkingen is hij te bereiken via Peter.van.Ooijen@Gekko-Software.nl

Nuttige internetadressen

- MSDN: http://msdn.microsoft.com/mobility/tabletpc/default.aspx
- Agilix: www. http://Agilix.com
- Weblog http://codebetter.com/blogs/peter.van.ooijen/archive/category/1047.aspx
- Algemene informatie: http://www.microsoft.com/windowsxp/tabletpc/default.mspx